

1. Sql Table Creation And Insertion Statements For All Tables.

Sql Codes For Creating The Database Tables (Casteel, 2015)

Students Table

Create Table Student(

Sid Varchar2(17) Primary Key,

Surname Varchar2(15),

Forenames Varchar2(15),

Address Varchar2(50),

Mobilephone Char(17),

Ccode Char(7),

Cyear Number(1),

Dob Date,

Constraint Check_Cyear Check(Cyear In (1,2,3,4,5));

Sessions Table

Create Table Sessions(

Scode Number(3) Primary Key,

Stype Varchar2(8),

Mcode Varchar2(6),

Tutor Varchar2(20),

Sdt Date,

Edt Date,

Room Varchar(10),

Constraint Check_Type Check(Stype In ('Lecture', 'Lab', 'Tutorial', 'Seminar')));

Attendance Table

Create Table Attendance(

Ano Number(5) Primary Key,

Sid Varchar2(17),

Scode Number(3),

Csdt Date,

Constraint Student_Fk Foreign Key(Sid) References Student(Sid),

Constraint Session_Fk Foreign Key(Scode) References Sessions(Scode)

Trigger To Determine Age If 18 And Above*/

Create Or Replace Trigger

Student_Age

Before Insert

On Student

For Each Row

Declare

Birthdate Date;

Begin

Select :New.Dob Into Birthdate From Dual;

If Add_Months(Birthdate,216) > Trunc(Sysdate)

```
Then Raise_Application_Error(-20000,'Student Is Under 18 Years!');
```

```
End If;
```

```
End;
```

```
Data Population*/
```

```
Students Table
```

```
Insert          Into          Student          Values('Doe_J','Doe','John','225,          Main  
Ave','07648734629','Comp101',2,'Jan-14-1996');
```

```
Insert          Into          Student          Values('Leaky_M','Leaky','Mary','123,          Green  
Street','07664598734','Tech101',3,'Dec-16-1999');
```

```
Insert          Into          Student          Values('Rajah_Evans','Rajah','Evans','22,North  
Ave','07657798765','Comp101',4,'Jun-20-1998');
```

```
Sessions Table
```

```
Insert Into Sessions Values(100,'Lecture','Co4703','Harry Spark',To_Char('12-Apr-2022 08:30:00',  
'Dd-Mon-Yyyy Hh:Mi:Ss'),
```

```
To_Char('12-Apr-2022 11:30:00', 'Dd-Mon-Yyyy Hh:Mi:Ss'),'Cm210');
```

```
Insert Into Sessions Values(101,'Lab','Co4703','Harry Spark',To_Char('14-Apr-2022 09:00:00',  
'Dd-Mon-Yyyy Hh:Mi:Ss'),
```

```
To_Char('14-Apr-2022 12:00:00', 'Dd-Mon-Yyyy Hh:Mi:Ss'),'Cm210');
```

```
Insert Into Sessions Values(102,'Lecture','Co4704','Alex Smith',To_Char('19-Apr-2022 14:00:00',  
'Dd-Mon-Yyyy Hh:Mi:Ss'),
```

```
To_Char('19-Apr-2022 16:30:00', 'Dd-Mon-Yyyy Hh:Mi:Ss'),'Cm215');
```

```
Insert Into Attendance Values (10001,'Doe_J',100,To_Char('12-Apr-2022 08:25:30', 'Dd-Mon-Yyyy  
Hh:Mi:Ss'));
```

Insert Into Attendance Values (10002,'Leaky_M',100,To_Char('12-Apr-2022 08:35:20', 'Dd-Mon-Yyyy Hh:Mi:Ss'));

Insert Into Attendance Values (10003,'Doe_J',101,To_Char('14-Apr-2022 09:25:30', 'Dd-Mon-Yyyy Hh:Mi:Ss'));

Insert Into Attendance Values (10004,'Leaky_M',101,To_Char('14-Apr-2022 08:55:20', 'Dd-Mon-Yyyy Hh:Mi:Ss'));

Insert Into Attendance Values (10005,'Rajah_Evans',102,To_Char('19-Apr-2022 14:05:20', 'Dd-Mon-Yyyy Hh:Mi:Ss'));

In The First Normal Form, The Table Contains Only Atomic Values And Each Record Is Unique. Below Is The Relation In The First Normal Form. The Primary Key Field Is Underlined (Coronel & Morris, 2016).

Booking (Bookingnumber, Bookingdate, Customercode, Customerlocation, Lorryregistrationno, Lorrytype,

Lorrycapacityintonnes,Driverlicenseno, Drivermobilephoneno)

Primary Key (Bookingnumber)

In Second Normal Form, The Table Must Be In 1Nf With a Single Column Primary Key That Does Not Functionally Depend On Any Subset Of Candidate Keys. Below Is a Set Of Tables Normalized To 2Nf With The Primary Keys Underlined And Foreign Keys Italicized.

Driver (Driverlicenseno, Drivermobilephoneno)

Primary Key (Driverlicenseno)

Lorry (Lorryregistrationno, Lorrytype, Driverlicenseno)

Primary Key (Lorryregistrationno)

Foreign Key (Driverlicenseno)

Booking (Bookingnumber, Bookingdate, Customercode, Customerlocation, Lorryregistrationno)

Primary Key(Bookingnumber)

Foreign Key (Lorryregistrationno)

In Third Normal Form, a Relation Must Be In 2Nf And With No Transitive Functional Dependencies. In This Case, Transitive Functional Dependencies Exist In The Booking Table (Harrington, 2016). Changing The Non-Key Column Customer Location May Need a Change In The Customer Code. Below Is a Set Of Database Tables Normalized To 3Nf. Primary Keys Underlined And Foreign Keys Italicized.

Driver (Driverlicenseno, Drivermobilephoneno)

Primary Key (Driverlicenseno)

Lorry (Lorryregistrationno, Lorrytype, Driverlicenseno)

Primary Key (Lorryregistrationno)

Foreign Key (Driverlicenseno)

Customer (Customercode, Customerlocation)

Primary Key (Customercode)

Booking (Bookingnumber, Bookingdate, Customercode, Lorryregistrationno)

Primary Key(Bookingnumber)

Foreign Key (Lorryregistrationno)

Foreign Key (Customercode)

Database Normalization Refers To The Process Of Modifying a Complex Database Into a Simpler Database Basically To Avoid Data Redundancy And Database Anomalies. The Following Are The Advantages And The Disadvantages Of Using Normalization As a Technique For Database Design.

1. Normalization Ensures Data Consistency – a Well Normalized Database Ensures That The Data Stored Is Real And Not Ambiguous. Consistent Records Will Be Stored In The Database. For Instance, If An Un Normalized Relation Is Used, a Lorry With a Particular Registration Number May Consist Of Inconsistent Details Of Its Driver.
2. There Is Reduced Data Redundancy – Normalization Ensures That There Are No Multiple Copies Of The Same Data, i.e. There Is No Data Repetition (Dietrich, 2021).For Instance,

Basing On Our Normalized Tables, Creating a Booking Record Will Not Require Repetitively Keying In All The Lorry, Driver And Customer Details. Instead, a References To These Tables Will Just Be Used.

3. There Is Reduced Database Anomalies – The Insertion, Deletion And Update Anomalies Which Often Arise When The Database Is Not Normalized Are Reduces. For Example, Updating Or Deleting All The Records With Lorry Type Panel Will Affect Many Records Than Expected. However, With Normalization, All The Database Anomalies Are Reduced And Prevented.
4. Database Table Compaction Is Achieved – Normalizing a Database Converts a Larger Table Into Smaller Tables That Achieves Table And Data Compaction. Compaction Ensures That The Database Has The Least And Required Size. For Instance, Normalizing The Booking Relation Will Ensure Small Tables Are Achieved And The Database Has a Smaller Size.
5. Normalization Enhances Better Performance – The Normalized Database Will Execute Faster Compared To The Un Normalized Database. Queries Written Against a Normalized Database Runs Faster And Efficient Leading To Better Performance.
1. An Experienced Database Designer Is Required – Understanding The Concept Of Database Normalization And Table Relationships Requires An Experienced Database Designer.
2. A Detailed Database Design Is Required – Normalization Requires a Detailed Database Design. This May Include Designing a Database That Lacks The Functional Dependencies With The Referential Integrity Constraints Observed.
3. Normalization Results Into a Difficult And Expensive Database – a Normalized Database Is Often Difficult And Expensive To Manage. For Instance, Retrieving Related Records Requires The Use Of Joins Which Are Often Expensive. For Instance, Retrieving Customers With Their Respective Booking Details Is Easier When The Data Is Un Normalized But Difficult And Expensive After Normalization As Joins Are Required In The Sql Select Statements.

References

Casteel, J., 2015. Oracle 12c: Sql. Boston, Ma: Cengage Learning.

Coronel, C. & Morris, ., 2016. Database Systems: Design, Implementation, & Management. Boston, Ma: Cengage Learning.

Dietrich, S. W., 2021. Understanding Databases: Concepts And Practice. Hoboken, New Jersey, U.S: John Wiley & Sons.

Harrington, J. L., 2016. Relational Database Design And Implementation. Amsterdam, Netherlands: Elsevier Science.